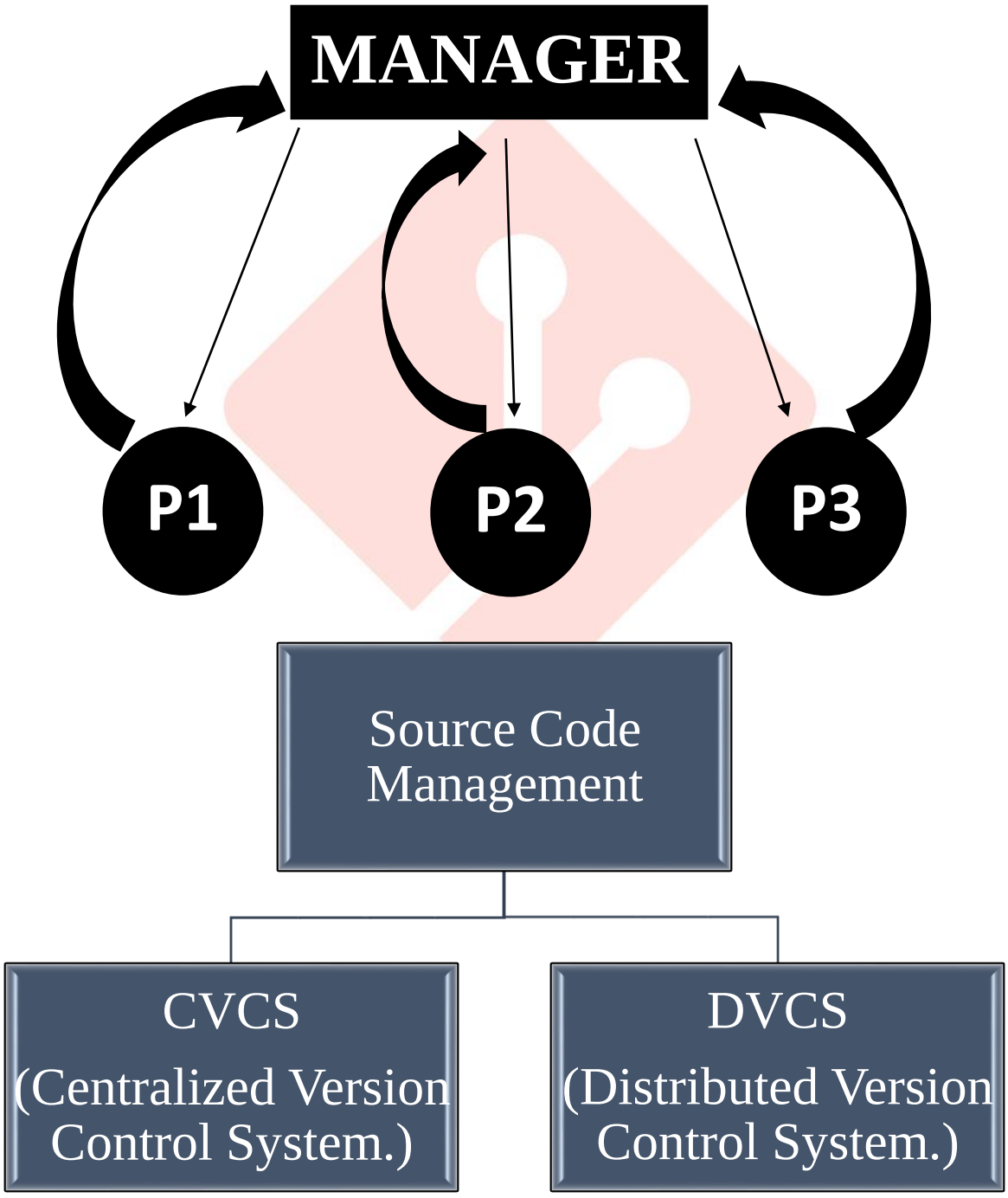
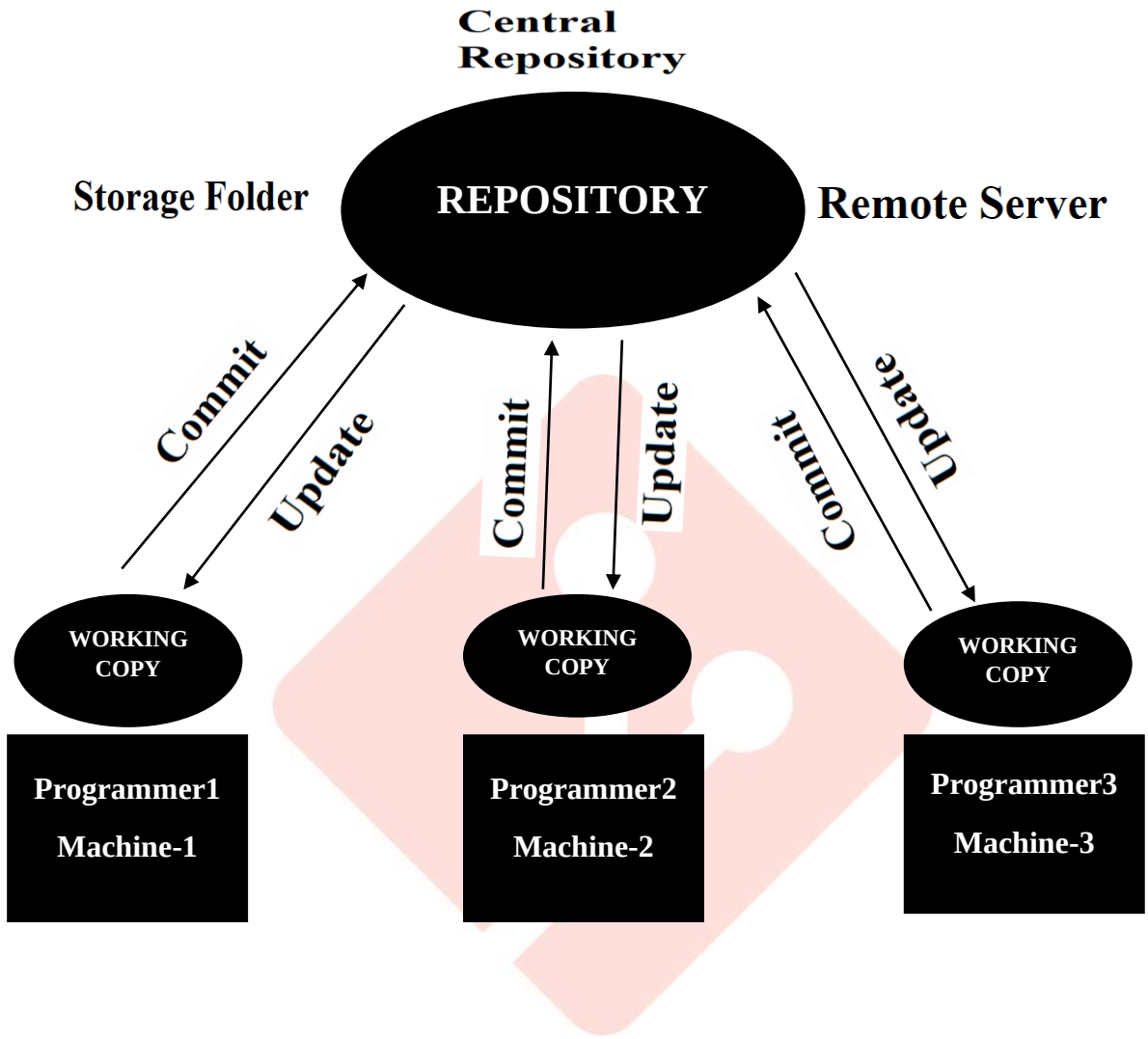
INTRODUCTION

Git is a **Software Configuration Management or Source Code Management (SCM)** tool. Which was developed by ***Linux Torvalds in 2005***. It is written on Linux.



varshneymohit35@gmail.com
Stepogrammer
8077636394

WORKING Of CVCS

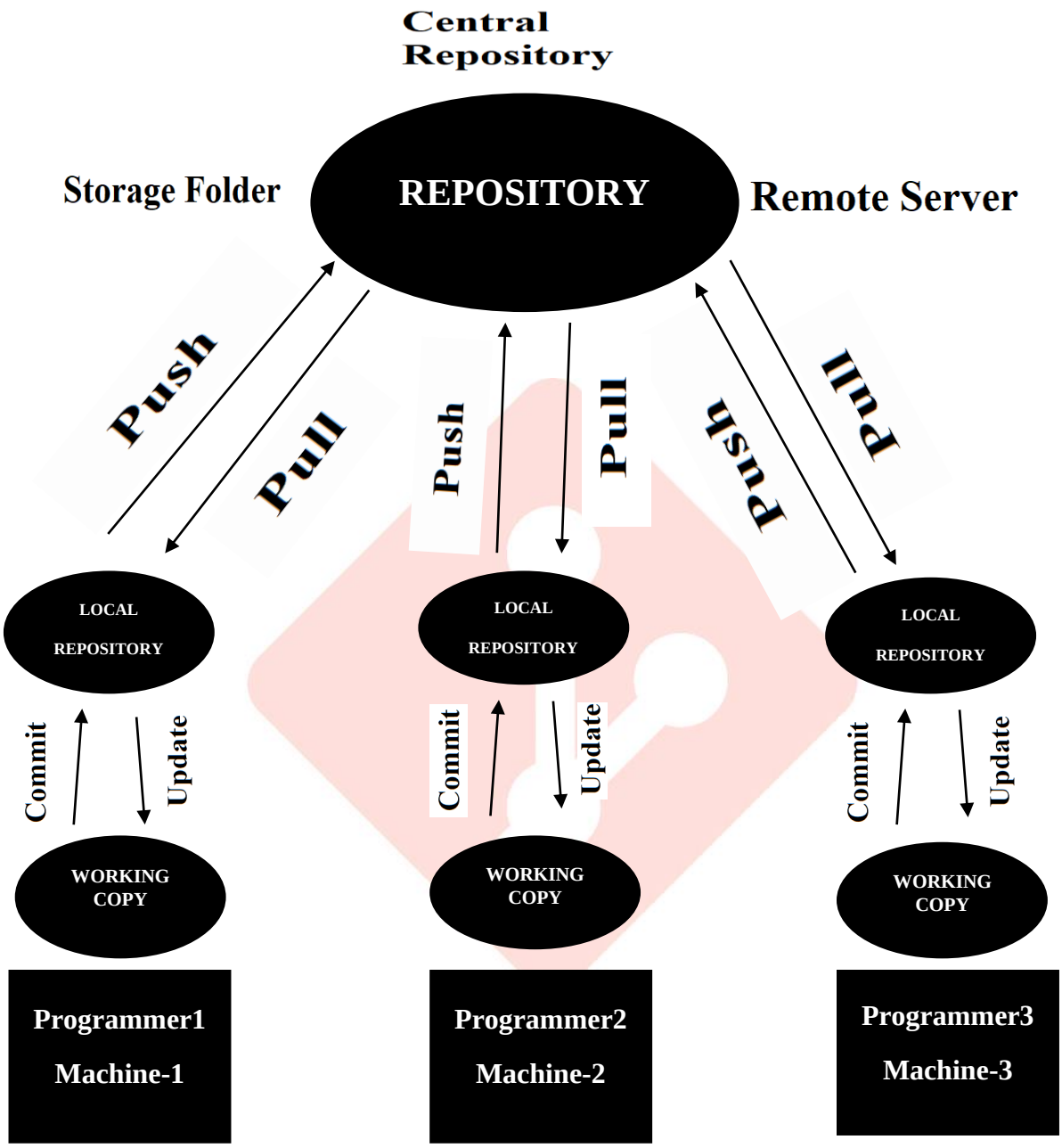


Disadvantages of CVCS (Centralized version control System):-

1. You need to connect with the Internet Every time to perform any changes in the repository. It is not locally available.
2. Central Repository gets fail down you will lose entire DATA eg. SVN(Subversion) tool.

varshneymohit35@gmail.com
Stepogrammer
8077636394

WORKING OF DVCS

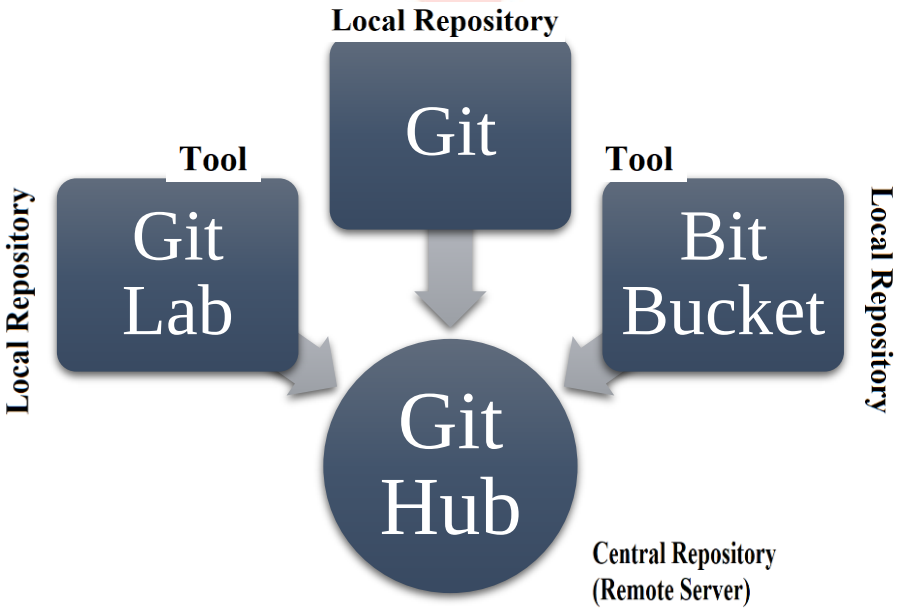


Note: In *DVCS* each contributor has its own local copy of the Main Repository that is maintain a separate repository which contains all the files and **Meta Data** information of the **Main/Central Repository**.

varshneymohit35@gmail.com
Stepogrammer
8077636394

Difference between CVCS and DVCS

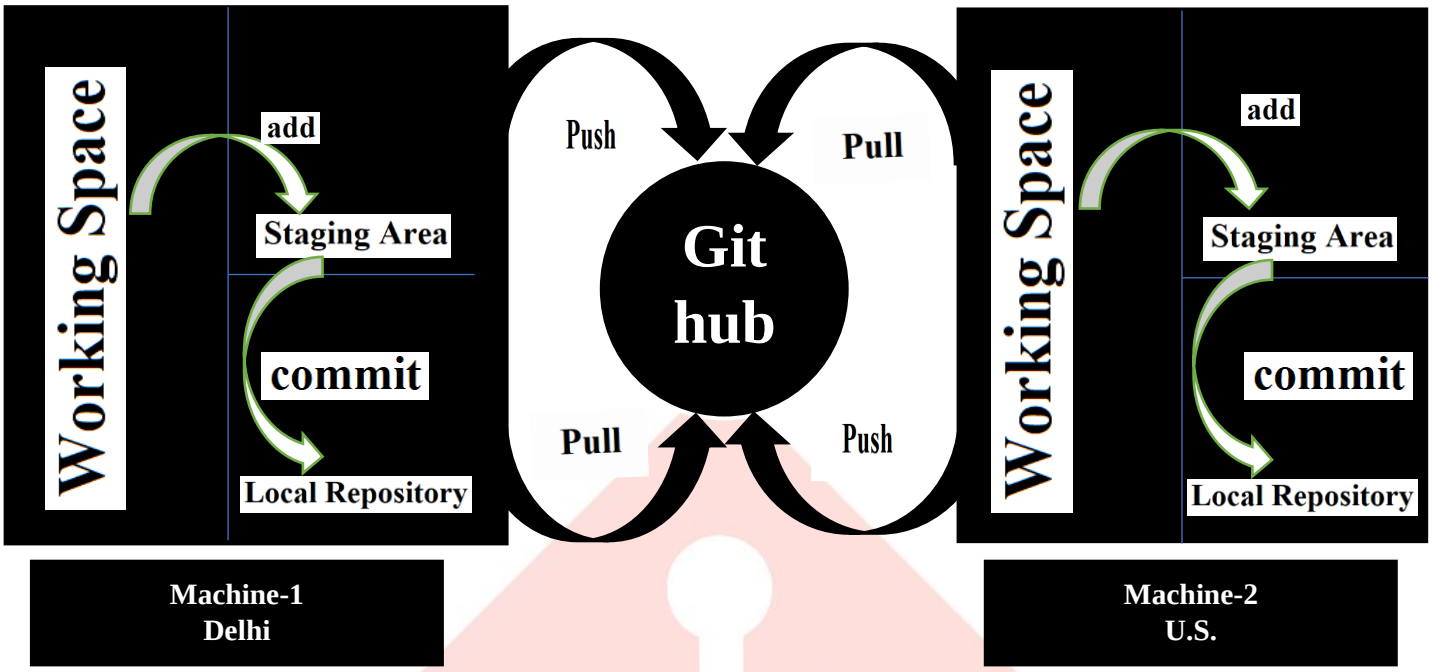
CVCS	DVCS
1. In CVCS a client needs to get local copy of source file, do not changes & commit these changes to central source on server.	In DVCS each client can have a local branch as well and have a complete history on it. Client need to push the changes to branch which will then be pushed to server repository.
2. CVCS system are easy to learn & setup.	DVCS systems are difficult for beginners' multiple commands need to be remembered.
3. Working on branches is difficult in CVCS, developer often faces merge conflict.	Working on branches is easier in DVCS, developer faces less difficulty.
4. CVCS system do not provide offline access.	DVCS systems are working fine on offline mode as a client copies the entire repository on the local machine.
5. CVCS is slower as every command need to be communicated with server.	DVCS is faster as mostly user deals with local copy without hitting server every time.
6. IF CVCS server is down developer can't work.	If DVCS server is down developer can work using their local copy.



varshneymohit35@gmail.com
Stepogrammer
8077636394

WORKING OF GIT

Three stage Architecture of Git:



Git Terminology

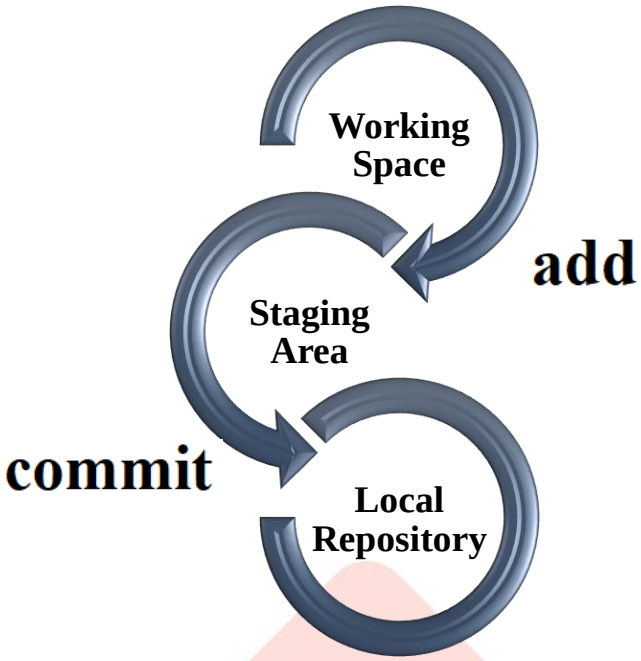
- **Repository:** Repository is a place where you have your all codes or kind of folder on server. It is a kind of folder related to one product. Changes are personal to that particular repository.

- **Server:** It stores all repositories.
  - It contains metadata also.

- **Working Directory:** Where you see files physically and do modifications. At a time, you can work on particular branch.

  Note: In other CVCS, developer generally make modification & commit their changes directly to the repository. but in GIT uses a different strategy, git does not track each n every modified file whenever you do commit an operation git looks for the files present in staging area are considered for commit & not all the modified files.

varshneymohit35@gmail.com
Stepogrammer
8077636394



- **Commit ID/Version ID/Version:** Reference to identify each change. TO identify who changed the file.

- **Tags:** Tags assign a meaningful name with a specific version in the repository once a tag is created for a particular save, even if you create a new commit it will not be updated.

- **Snapshots:** Represents some data of particular time. It is always incremental i.e. it stores the changes (appended data) only not entire copy.

- **Commit:**

  - Store changes in repos, you will get on commit-id.

  - It is 40digit alphanumeric characters.

  - It uses SHA-1 checksum concept.

  - Even if you change one dot commit-id will get change.

  - Commit is also named as SHA1 hash.

varshneymohit35@gmail.com
Stepogrammer
8077636394

- **PUSH:** Push operation copies changes from a local repo instances to a remote or central repo. This is used to store the changes permanently into the git repo.

- **PULL:** Pull operation copies the changes from a remote repo to a local machine. The pull operation is used for synchronization between two repos.

- **Branch:**

  - Product is same so one repo but different task.

  - Each task has one separate branch.

  - Finally merge(codes) all branches.

  - useful when you want to work parallel.

  - Can create one branch on the basis of another branch.

  - Changes are personal to that particular branch.

  - Default branch is master.

  - File created in workspace will be visible in any of the branch workspace until you commit, once you commit then that file belongs to that particular branch.

- **Advantages of GIT:**

  - Free & open source.

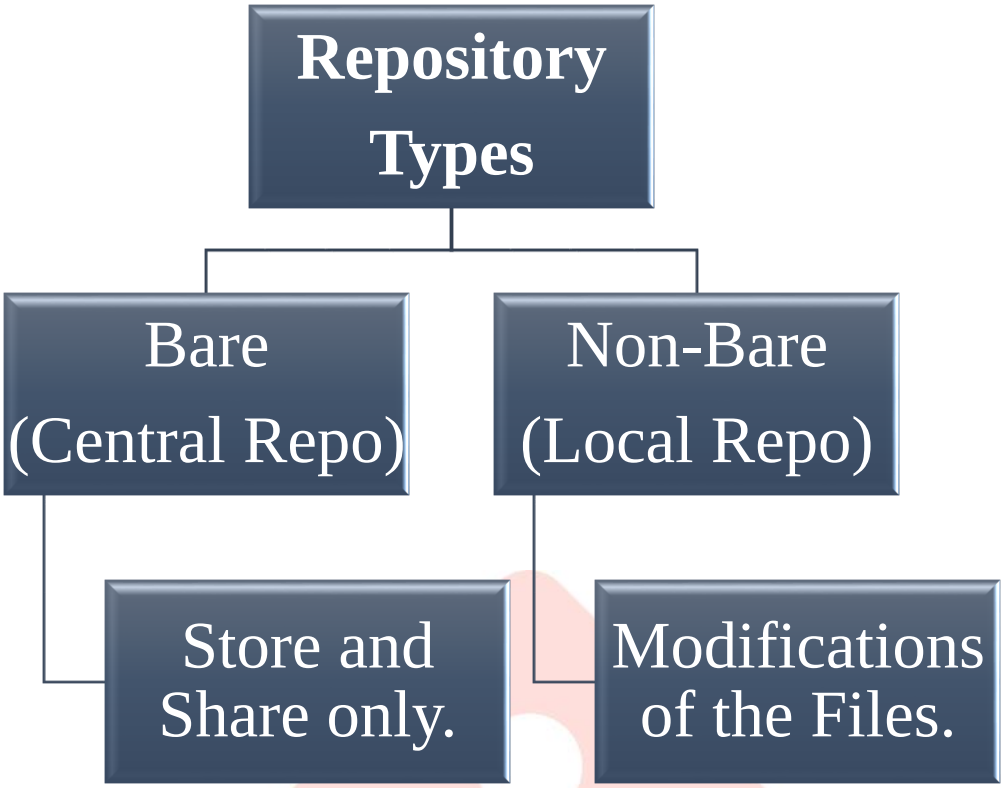
  - Fast & small- as most of the operations are performed locally therefore it is fast.

  - Security- git uses a common cryptography hash function called Secure Hash Function (SHA-1) to name & identify objects within its database.

  - No need of powerful hardware.

  - Easier branching: if we create a new branch it will copy all the code to the branch.

varshneymohit35@gmail.com
Stepogrammer
8077636394



Git Branching/Merge Branching

1-This diagram visualize a repository with two isolated lines of development.

2- One for little feature and One for a longer running feature by developing them in branches it is not only possible to work on.

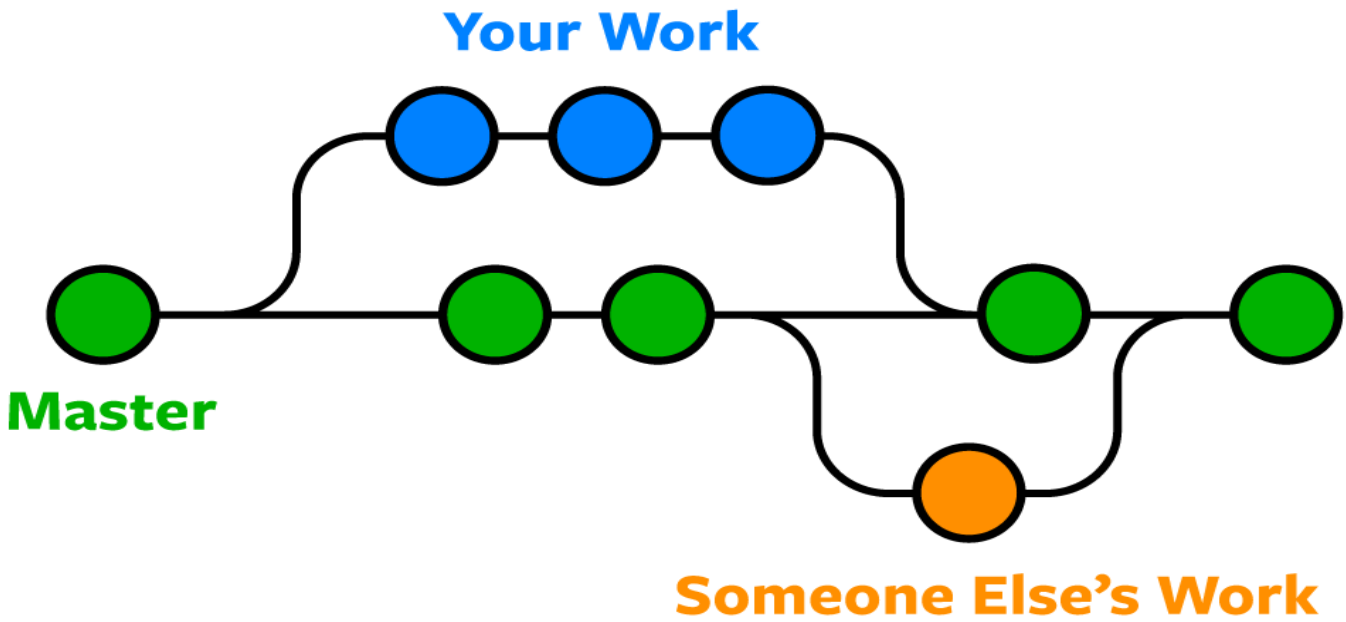
3- Both of them parallel but it keeps also a main master branch, free from errors.

Note:

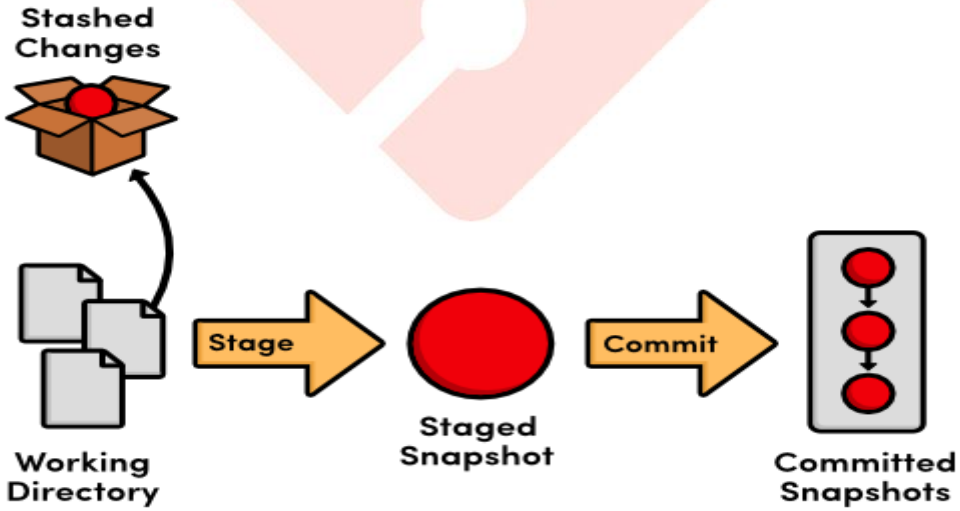
1- each task has one separate branch.
2- after done with code merge other branches in master.
3- this concept is useful for parallel development.
4- you can create any no. of branches.
5- changes are personal to dead particular branch.
6- default branch is master branch.

varshneymohit35@gmail.com
Stepogrammer
8077636394

7- files created in work space will be visible in any no. of th branch workspace, until you commit.
once you commit then dead files belongs to the particular branch.
8- when create a new branch data of the existing branch is copy to the new branch.



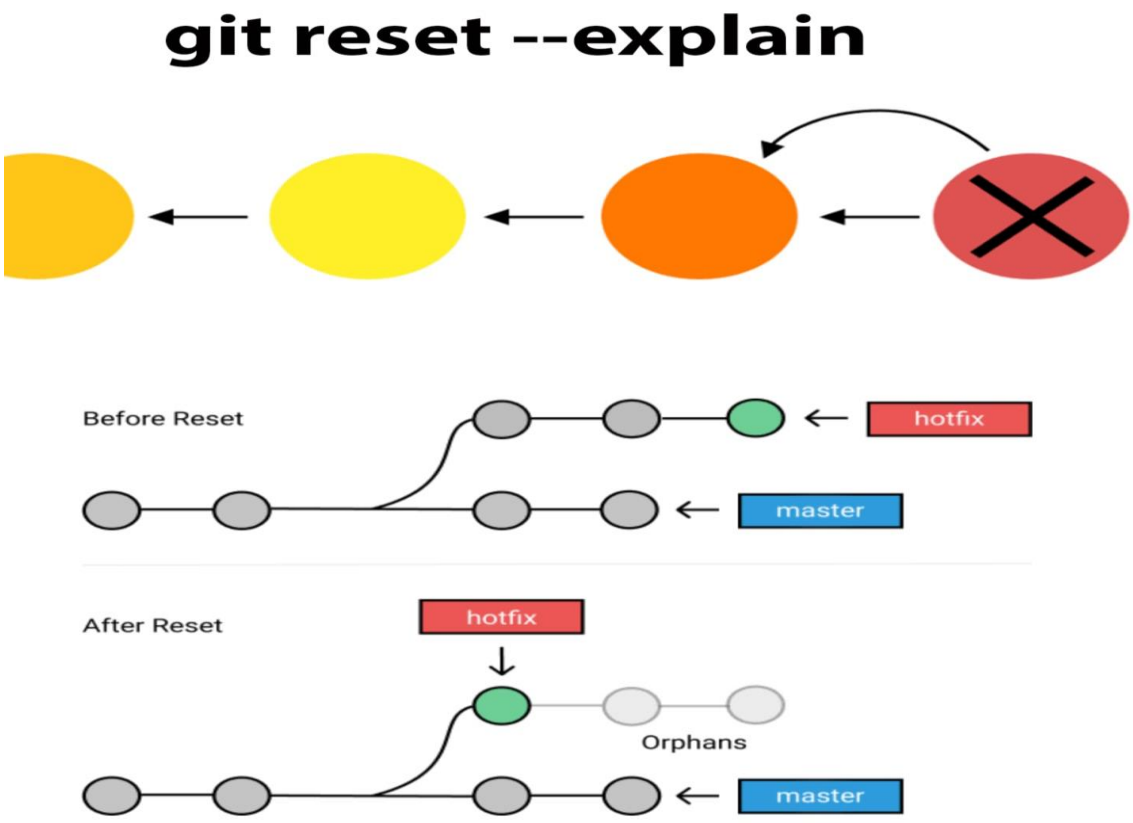
Git Stashing: It is used to put workspace area code into temporarily memory which is called stash.



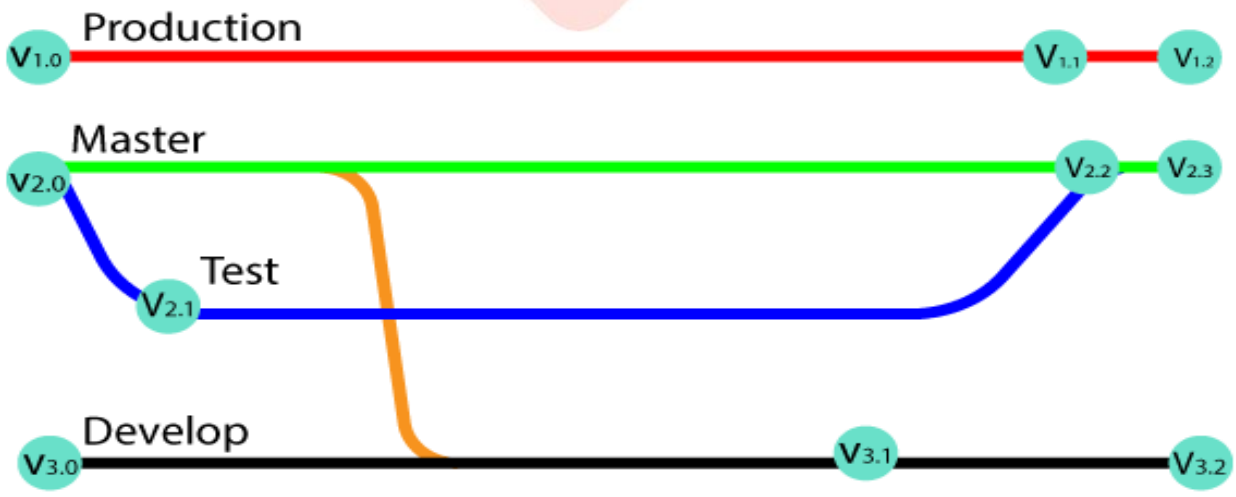
GIT RESET: it is used to undo the changes from staging area to workplace.

varshneymohit35@gmail.com
Stepogrammer
8077636394

GIT REVERT: it is used to undo the last commit.



GIT TAG: it is used to assign tags to the commits.



varshneymohit35@gmail.com
Stepogrammer
8077636394

Git Commands

Commands	Work
git init	Git initialize in directory
sudo su	Creating superuser.
git --version	Check git version.
git status	Check git status.
git config --global	Git configuration globally.
git config --global user.email“e-id”	Git email configuration.
git config --list	Check list.
touch <filename>	For create a new file.
Vi	For open editor
I	For insert data in file
Esc+:+wq+enter	For exit from editor console.
cat > “file name”	Create file with editor mode.
cat “filename”	Open file
Ctrl+d	For save data in file.
Ctrl+L	For clear screen.
git add .	For all files send to staging area.
git add “filename”	For specific file adding in staging area.
git commit -m “commit-message”	For commit file.
git log	For check commits with commit id.
git log --oneline	For check commits in one line.
git show “commit id”	For display specific commit.
git remote add origin <central-repo url>	For establish connection with central-repository.
git push -u origin master	For **Push** data/file to central repository.
git pull origin master.	For **Pull** data/file from central repository.
LS (small)	For checking file list in directory.
*** with extension**	For exclude a file in pull or push. For this task use **.gitignore** file as a exclude extensions.
git branch (branch name)	For create another branch.
git checkout (branch name)	For switch to another branch.
git merge "Branch name"	For merging branch to another/master branch.
git stash	For file sent to stashing area.
git stash list	For check stashing file list.

varshneymohit35@gmail.com
Stepogrammer
8077636394

git stash apply <stash-id>	For pull file from stashing area to working space.
git stash clear	For clear stash area.
git reset "filename"	For reset file from staging area before commit.
git reset --hard	For reset file from all git environment
git revert	For revert after commit.
git tag -a tagname -m message commitid	For tagging to commits for easily find and use in future.
git tag	For display all tags.
git show tagname	For display specific tags as per choice.
git tag -d tagname	For deleting specific tag.
git clean -n	For delete file before add in staging area.
git clean -f	For forcefully delete from working space.
git clone <central-repo-url>	For making clone of central repository for other requirements.

varshneymohit35@gmail.com
Stepogrammer
8077636394